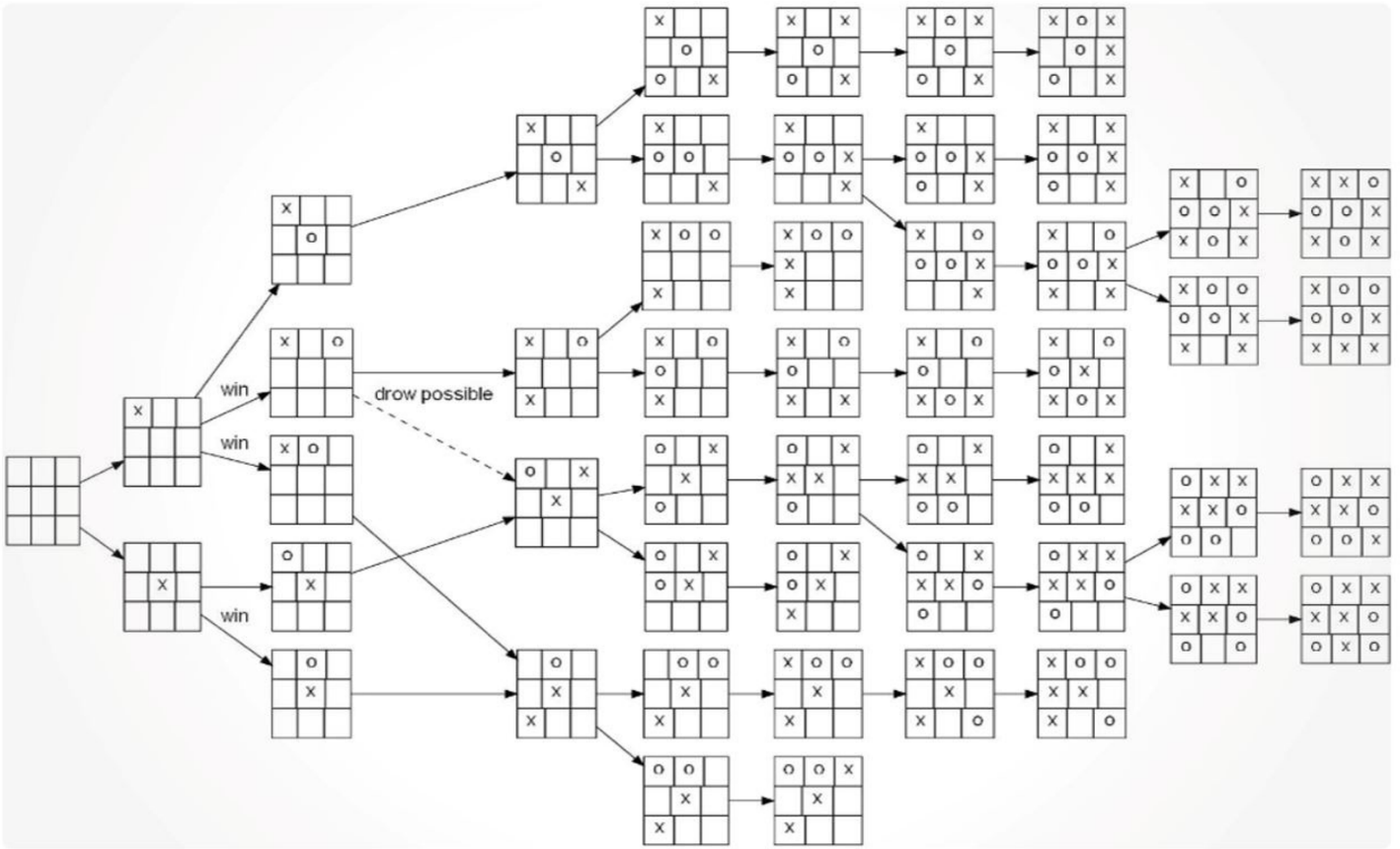
СПБГЭУ UNECON

КРЕСТИКИ НОЛИКИ

ФАЛЬКОВ КОНСТАНТИН ПМ-2101

О ПРОЕКТЕ



ПОПЫТКИ

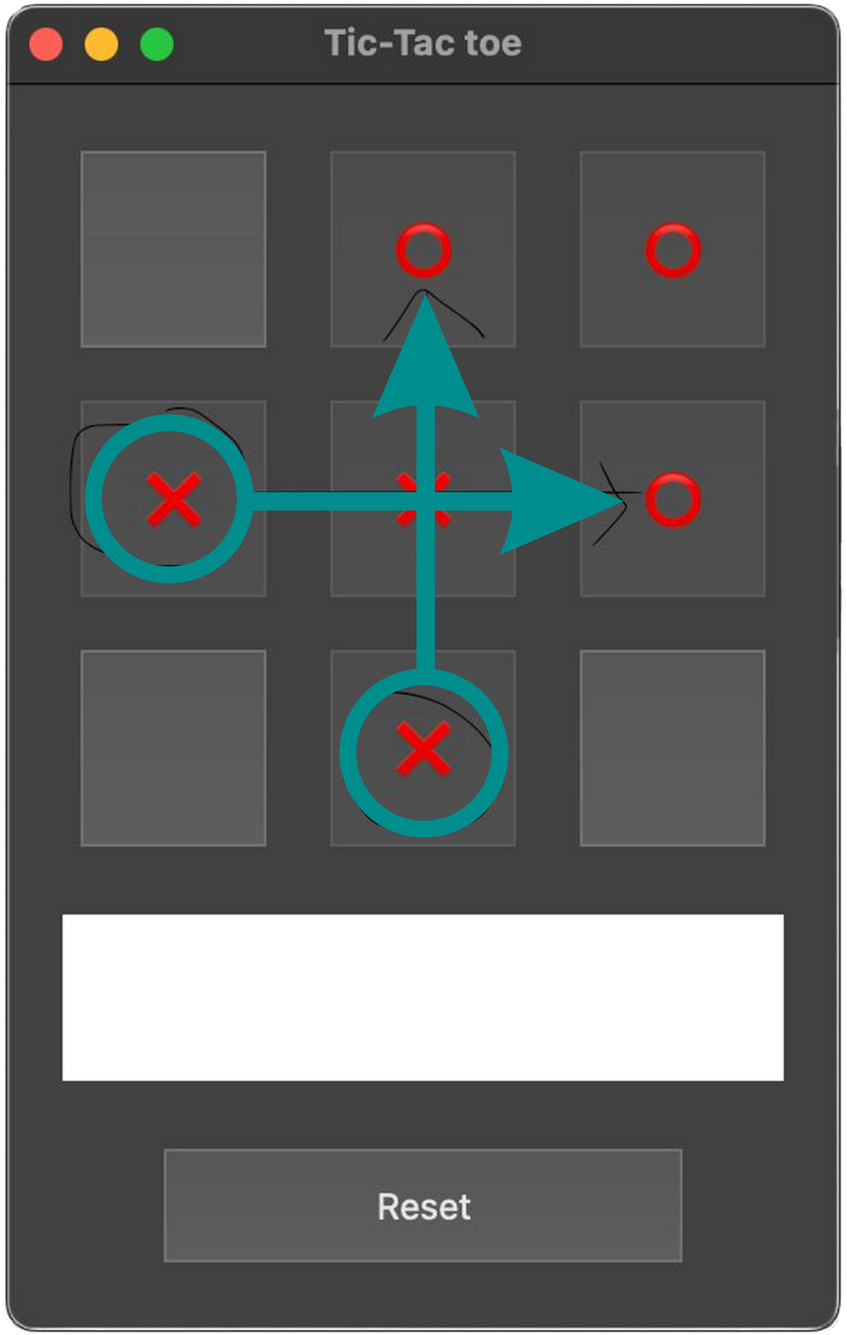
НЕУДАЧНАЯ ПОПЫТКА ПРОСЧИТАТЬ ВСЕ КОМБИНАЦИИ

```
# if step1 == 2:
#     self.field_buttons[0][2].click()
#     # [0 0 o]
#     # [0 x 0]
#     # [0 0 0]
#     flag = 0
#     if self.field[0][0] == 1 and flag != 1:
#         self.field_buttons[2][2].click()
#     if self.field[0][1] == 1:
#         self.field_buttons[1][2].click()
#     if self.field[0][2] == 1:
#         self.field_buttons[0][2].click()
#     if self.field[1][2] == 1:
#         self.field_buttons[0][1].click()
#     if self.field[2][0] == 1:
#         self.field_buttons[2][0].click()
#     if self.field[2][1] == 1:
#         self.field_buttons[1][0].click()
#     if self.field[2][2] == 1 and flag != 1:
#         self.field_buttons[0][0].click()
#     if self.field[1][0] == 1:
#         self.field_buttons[2][1].click()
#     if self.field[0][2] == 1:
#         delicateStep = random.randrange(0, 2)
#         if delicateStep == 1:
#             flag = 1
#             self.field_buttons[2][2].click()
#         else:
#             flag = 1
#             self.field_buttons[0][0].click()
```

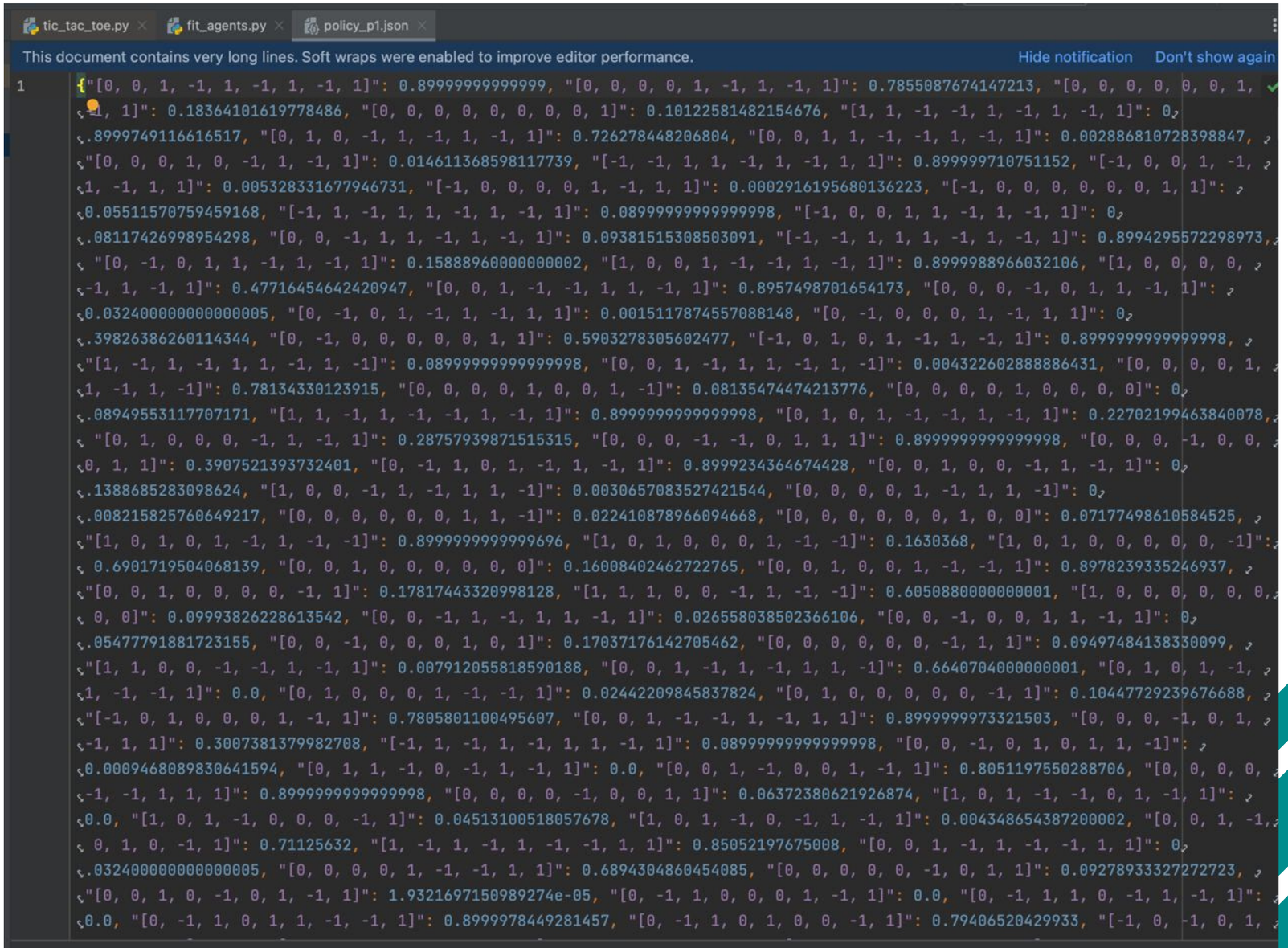
ПЕРВЫЕ УСПЕХИ

```
if self.turn % 2 + 1 == 2:
    self.field[x][y] = -1
else:
    self.field[x][y] = 1
```

ПЕРВАЯ КОМБИНАЦИЯ



REINFORECMENT LEARNING



```python
def checkWinner(self):
    """
    Функция определяет победителя в игре
    :return: 1 - победили крестики, -1 - нолики, 0 -ничья
    """
    # Смотрим по строкам
    for i in range(3):
        row_sum = sum(self.field[i])
        if row_sum == 3:
            self.isEnd = True
            return 1
        if row_sum == -3:
            self.isEnd = True
            return -1

    # Смотрим по столбцам
    for i in range(3):
        col_sum = sum([self.field[k][i] for k in range(3)])
        if col_sum == 3:
            self.isEnd = True
            return 1
        if col_sum == -3:
            self.isEnd = True
            return -1

    # Смотрим по диагоналям
    diag_sum1 = sum([self.field[i][i] for i in range(3)])
    diag_sum2 = sum([self.field[i][3 - i - 1] for i in range(3)])
    diag_sum = max(abs(diag_sum1), abs(diag_sum2))
    if diag_sum == 3:
        self.isEnd = True
        if diag_sum1 == 3 or diag_sum2 == 3:
            return 1
        else:
            return -1

    # Если ничья
    if len(getAvailablePositions(self.field)) == 0:
```

О РАБОТЕ ОБУЧЕНИЯ

ФУНКЦИЯ ПРОВЕРКИ ПОБЕДИТЕЛЯ

```python
def getAvailablePositions(field):
    """
    Определение списка из возможных позиций
    :return: список ячеек, куда можно поставить крестик или нолик
    """
    positions = []
    for i in range(3):
        for j in range(3):
            if field[i][j] == 0:
                positions.append((i, j))
    return positions
```

О РАБОТЕ ОБУЧЕНИЯ

ПОИСК НЕЗАНЯТЫХ ПОЛЕЙ

```
def getHash(field):
    """
    Создаёт хэш состояния игры
    :return: строка из массива с полем игры
    """
    state = []
    for row in field:
        for cell in row:
            state.append(cell)

    return str(state)
```

О РАБОТЕ ОБУЧЕНИЯ

СОХРАНЯЕМ СОСТОЯНИЕ ИГРЫ

```python
def giveReward(self):
    """
    Раздать агентам награду после игры
    """
    result = self.checkWinner()
    if result == 1:
        self.tic_player.feedReward(1)
        self.tac_player.feedReward(0)
    elif result == -1:
        self.tic_player.feedReward(0)
        self.tac_player.feedReward(1)
    else:
        self.tic_player.feedReward(0.1)
        self.tac_player.feedReward(0.5)
```

О РАБОТЕ ОБУЧЕНИЯ

ПООЩРЕНИЕ ИГРОКА

ПОЛУЧАЕМ ВОЗМОЖНЫЕ ПОЗИЦИИ,СОСТОЯНИЕ ДОСКИ И ЧЬЯ ОЧЕРЕДЬ ДЛЯ ХОДА

С ВЕРОЯТНОСТЬЮ 30% СЛЕДУЮЩИХ ХОД РАНДОМНЫЙ

ПОЛУЧАЕМ VALUE ПООЩРЕНИЯ ДЛЯ СЛЕДУЮЩИХ ХОДОВ, ДЕЛАЕМ НОВУЮ ДОСКУ,ХЭШИРУЕМ ЕЕ И СРАВНИВАЕМ РЕЗУЛТАТЫ ОТ ХОДОВ -> ИЩЕМ МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ И СОХРАНЯЕМ НУЖНОЕ ДЕЙСТВИЕ

```python
def chooseAction(self, positions, current_board, symbol):
    """
    Выбор действия
    :param positions: возможные клетки, куда можно поставить символ
    :param current_board: состояние поля
    :param symbol: какой символ нужно поставить
    :return:
    """
    # С заданной вероятностью действие случайное
    if random.random() <= self.exp_rate:
        action = random.choice(positions)
    else:
        action = [-1, -1]
        value_max = float('-inf')
        for p in positions:
            next_board = copy.deepcopy(current_board)
            next_board[p[0]][p[1]] = symbol
            next_boardHash = getHash(next_board)
            if self.states_value.get(next_boardHash) is None:
                value = 0
            else:
                value = self.states_value.get(next_boardHash)

            if value >= value_max:
                value_max = value
                action = p
    return action
```

```
def feedReward(self, reward):
    """
    Обновить значения для всех состояний, которые были в игре, в зависимости от её исхода
    :param reward: награда после игры
    """
    for state in reversed(self.states):
        if self.states_value.get(state) is None:
            self.states_value[state] = 0
        self.states_value[state] += self.lr * (self.decay_gamma*reward - self.states_value[state])
        reward = self.states_value[state]
```

ОБНОВЛЯЕМ ЗНАЧЕНИЕ ВСЕХ СОСТОЯНИЙ В ЗАВИСИМОСТИ ОТ НАГРАДЫ

ПОКА ИГРА НЕ ЗАКОНЧЕНА- ДОСТАЕМ ВСЕ ПОЗИЦИИ, ИЗУЧАЕМ ВОЗМОЖНЫЕ СЛЕДУЮЩИЕ ХОДЫ, ПРОВЕРЯЕМ НА ЗАВЕРШЕНИЕ ИГРЫ И ВЫДАЕМ НАГРАДЫ, ОБНОВЛЯЕМ СОСТОЯНИЯ И ОЧИЩАЕМ ДОСКУ. (ОДИНАКОВО ДЛЯ КРЕСТИКА И НОЛИКА)

```python
def fit(self, rounds):
    for i in range(rounds):
        if i % 1000 == 0:
            print(f'Игра №{i}')
        while not self.isEnd:
            # Ход крестиков
            positions = getAvailablePositions(self.field)
            tic_cell = self.tic_player.chooseAction(positions, self.field, self.playerSymbol)
            # Обновляется состояние
            self.updateState(tic_cell)
            field_hash = getHash(self.field)
            self.tic_player.states.append(field_hash)

            # Определяем победил ли игрок
            win = self.checkWinner()
            if win is not None:
                # self.showBoard()
                # ended with tic_player either win or draw
                self.giveReward()
                self.tic_player.reset()
                self.tac_player.reset()
                self.reset()
                break

            else:
                # Ход ноликов
                positions = getAvailablePositions(self.field)
                tac_action = self.tac_player.chooseAction(positions, self.field, self.playerSymbol)
                self.updateState(tac_action)
                field_hash = getHash(self.field)
                self.tac_player.states.append(field_hash)

                # Победил ли игрок
                win = self.checkWinner()
                if win is not None:
                    self.giveReward()
                    self.tic_player.reset()
```

ИСТОЧНИКИ

1.**REINFORCEMENT LEARNING — IMPLEMENT TICTACTOE**
HTTPS://TOWARDSDATASCIENCE.COM/REINFORCEMENT-LEARNING-IMPLEMENT-TICTACTOE-189582BEA542
2.**REINFORCEMENT Q-LEARNING FROM SCRATCH IN PYTHON WITH OPENAI GYM**
HTTPS://WWW.LEARNDATASCI.COM/TUTORIALS/REINFORCEMENT-Q-LEARNING-SCRATCH-PYTHON-OPENAI-GYM/
3. **HANDS ON INTRODUCTION TO REINFORCEMENT LEARNING IN PYTHON**
HTTPS://TOWARDSDATASCIENCE.COM/HANDS-ON-INTRODUCTION-TO-REINFORCEMENT-LEARNING-IN-PYTHON-DA07F7AACA88

СПАСИБО ЗА ВНИМАНИЕ!